Example of Tuning Cube II

Date: Dicember 2016

Author: Ramón Portolés, Alberto Linkedin a.ramonportoles@gmail.com

Tuning of M-R Map-Reduce

After tuning of cube: (See KylinPerformance_I)

- Hive Input tables compressed
- HBase Output compressed
- Apply techniques of reduction of cardinality (Joint, Derived, Hierarchy and Mandatory)
- Personalize Dim encoder for each Dim and Choose best order of Dim in Row Key

Now we have three types of cubes:

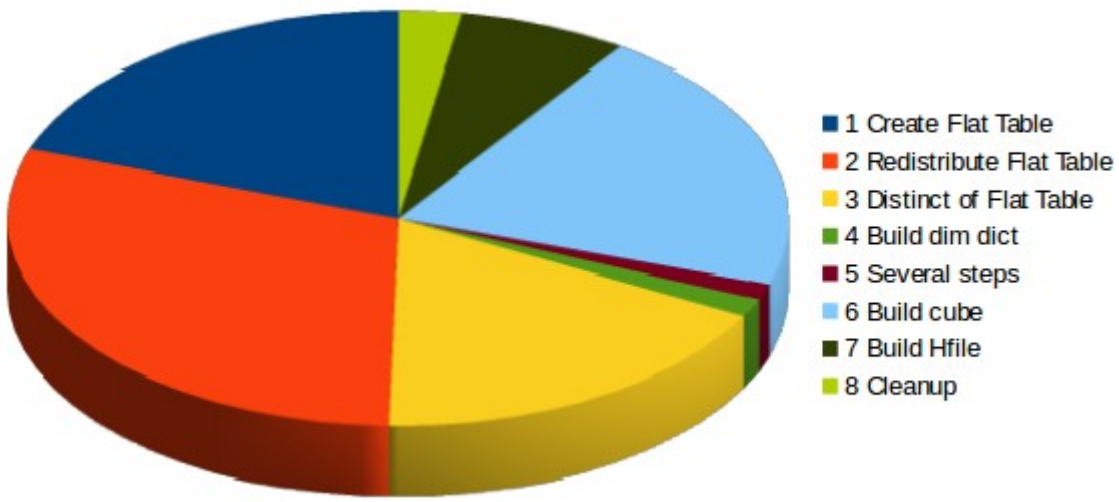
- Cubes with low cardinality in their dimensions
- Cubes with high cardinality in their dimensions
- The third type, ultra high cardinality (UHC) which is outside the scope of this article

Cubes with low cardinality Dimensions:

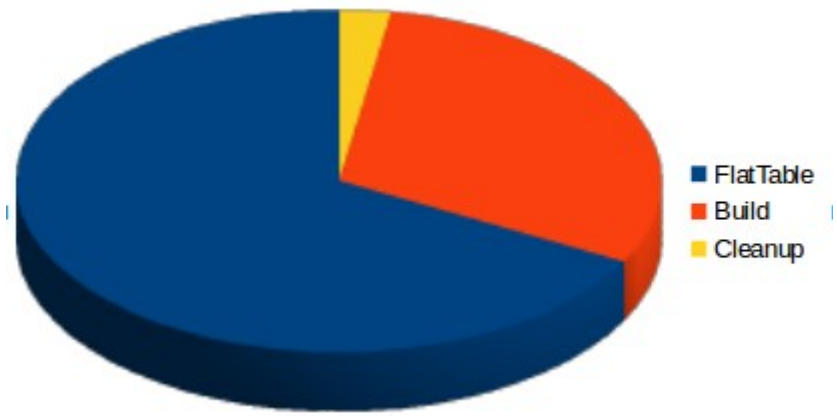
In our example:

- Fact table 4Millons of rows
- Dim1 2K & Dim2 12K of rows

Out Time Chart must be similar to this:



If we make a group by concept:



We can see the 67% of time is used on Flat Table (Steps 1,2,3)

Attempt 1:

How can try to reduce the time in these steps?

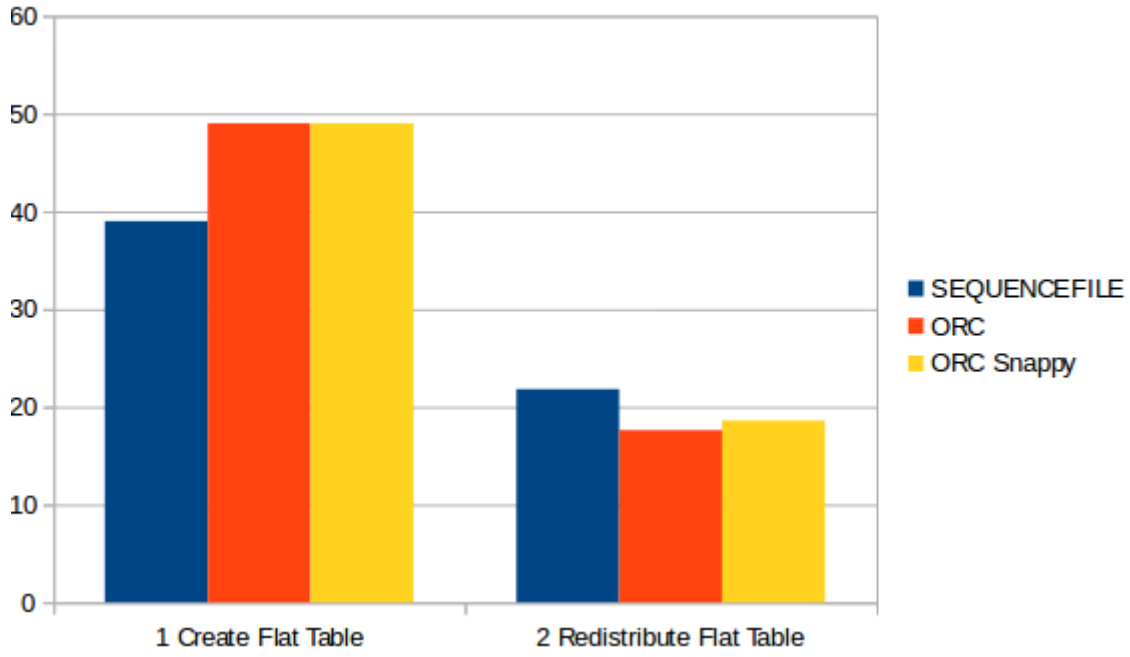
You can see the SQL Hive of these steps on Monitor >

```
USE default;
DROP TABLE IF EXISTS kylin_intermediate_Her_Position_Cube_06_d2cc2f0
CREATE EXTERNAL TABLE IF NOT EXISTS kylin_intermediate_Her_Position_
(
HERR_POSITIONS_FACT_POSICIONES2_ID_FECHA date
. . . .
,HERR_POSITIONS_FACT_POSICIONES2_POSICIONES decimal(28,8)
)
STORED AS SEQUENCEFILE
LOCATION '/kylin/kylin_metadata/kylin-bf409f37-289f-4a65-86bd-15a857
SET dfs.replication=2;
SET hive.exec.compress.output=true;
SET hive.auto.convert.join.noconditionaltask=true;
SET hive.auto.convert.join.noconditionaltask.size=100000000;
SET mapreduce.output.fileoutputformat.compress.type=BLOCK;
SET mapreduce.job.split.metainfo.maxsize=-1;
INSERT OVERWRITE TABLE kylin_intermediate_Her_Position_Cube_06_d2cc2
FACT_POSICIONES2.ID_FECHA
 . . .
,DIM_FECHAS2.ANYO
,FACT_POSICIONES2.POSICIONES
FROM HERR_POSITIONS.FACT_POSICIONES2 as FACT_POSICIONES2
INNER JOIN HERR_POSITIONS.DIM_FECHAS2 as DIM_FECHAS2
ON FACT_POSICIONES2.ID_FECHA = DIM_FECHAS2.ID_FECHA AND FACT_POSICIO
.
```

We can see:

- Uses external Tables of Hive
- Uses Sequencefile as storage format

We can try to use other columnar formats



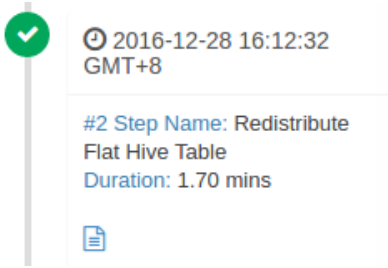
- ORC
- ORC compressed with Snappy

But the result are worst than Sequence file …

See comments about this from Shaofengshi in MailList

Attempt2:

The second strep (redistribute Flat Hive table)



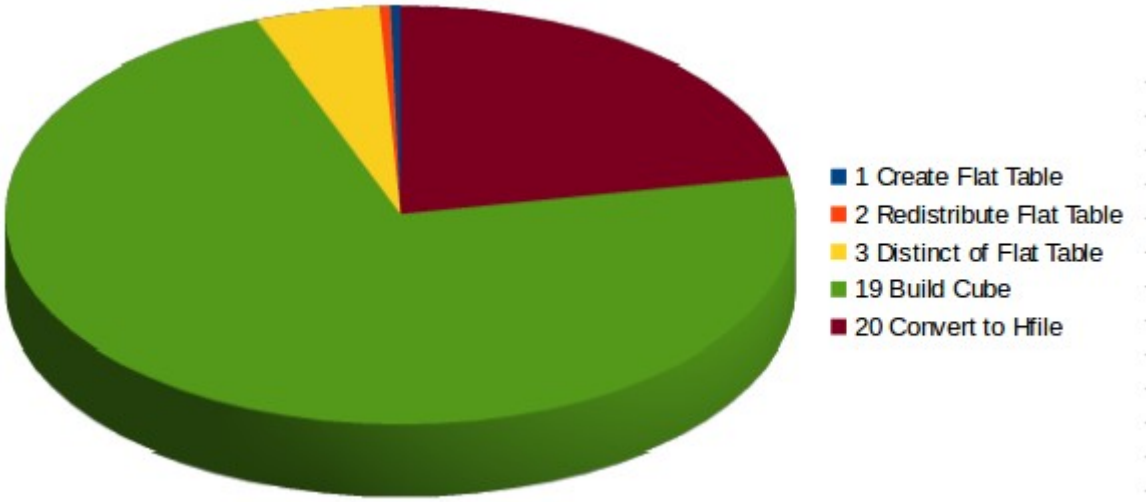
Is a simple row count:

```
set hive.exec.compress.output=false;
INSERT OVERWRITE DIRECTORY '/kylin/kylin_metadata/kylin-3b/row_count'
  SELECT count(*) FROM kylin_intermediate_Her_Position_Cube_11_;
```

We can think in some approx: If we don't need accurate, we can make a count of fact table → this can be performed in parallel with Step 1 (and 99% of times will be accurate)

See comments about this from Shaofengshi in MailList , In future version (Kylin 2265 v2.0) this steps will implemented using Hive table statistics

Cubes with high cardinality Dimensions:

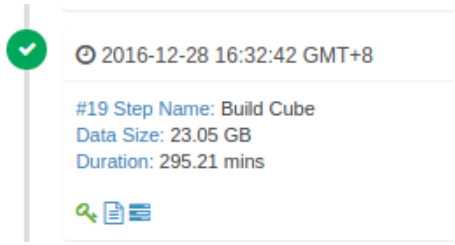


In this case the 72% of time is used to Build Cube

This step is a MapReduce task, you can see the Sql Hive of these steps on Monitor >

Attempt1:

How can improve the performance of Map – Reduce? The easy Way is increase the numbers of mappers and reduces (= Increase parallelism).



NOTE: YARN / MapReduce have a lot parameters to configure and adapt to your system, we only will be focus on small part of them

(In my system I can assign to YARN Resources: 12 – 14 GB and 8 cores)

- yarn.nodemanager.resource.memory-mb = 15 GB
- yarn.scheduler.maximum-allocation-mb = 8 GB
- yarn.nodemanager.resource.cpu-vcores = 8 cores

With this config our max teorical grade of paralelist is 8 , but this have a problem: “Timed out after 3600 secs”

Attempt	State	Elapsed Time	Note
attempt_1482868137604_0009_m_000000_0	FAILED	1hrs, 57sec	AttemptID:attempt_1482868137604_0009_m_000000_0 Timed out after 3600 secs
attempt_1482868137604_0009_m_000000_1	FAILED	1hrs, 56sec	AttemptID:attempt_1482868137604_0009_m_000000_1 Timed out after 3600 secs

The parameter *mapreduce.task.timeout* (1 hour by default) define max time that Application Master (AM) can happen with out ACK of Yarn Container. Once past this time, AM kill the container and retry the same 4 times (with same result)

Where is the problem? Our problem is that I starter 4 mappers, but each mapper need more than 4 GB to finish

The solution 1: add more RAM to YARN

The solution 2: add more Reduce the number to vCores

The solution 3: You can play with max RAM to YARN by node (*yarn.nodemanager.resource.memory-mb*) and min RAM to container (*yarn.scheduler.minimum-allocation-mb*). If you increase minimum RAM per container, YARN will reduce the numbers of maps in

Show 20 entries

Task	Progress	
task_1482877261039_0005_m_000000		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000001		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000002		hdfs://amb0.m 72d53d0ee2d3 > map
task_1482877261039_0005_m_000003		NEW

In three cases the result are the same: reduce the level of parallelism ==>

- Now we only start 3 mappers at same time, the fourth must be wait to free slot
- The three first mappers spread the ram, then they will have enough ram to finish the task

During a normal “Build Cube” step you must see similars messages on YARN log:

```
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 53059 has 948266 rows, build takes 12196ms
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:53059
[CuboidTask-2] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 52995
[CuboidTask-2] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 118531 has 955928 rows, build takes 12667ms
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:118531
[CuboidTask-3] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 53057
[CuboidTask-3] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Cuboid 118593 has 938836 rows, build takes 11976ms
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: collecting CuboidResult cuboid id:118593
[CuboidTask-0] org.apache.kylin.cube.inmemcubing.InMemCubeBuilder: Calculating cuboid 53058
[CuboidTask-0] org.apache.kylin.gridtable.GTScanRequest: pre aggregating results before returning
```

If you don’t see periodically, perhaps you have a bottleneck in memory

Attempt2:

We can try to use differences aggregations groups to improve the query performance of some Dim very important or with high cardinality.

In our case we define 3 Aggregations Groups:

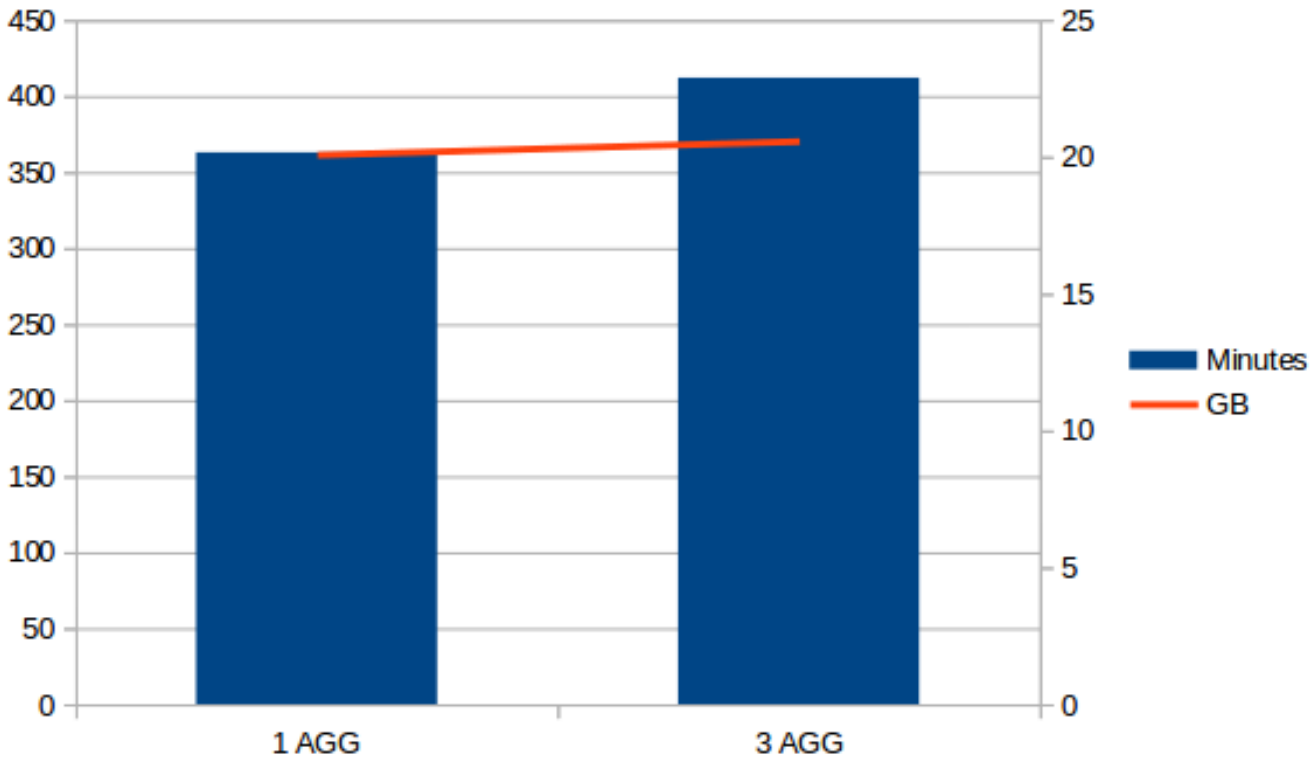
1. “Normal cube”
2. Cube with Date Dim and Currency (as mandatory)
3. Cube with Date Dim and Carteras_Desc (as mandatory)

ID	Aggregation Groups	
1	Includes	["COD_PRODUCTO"," PO_PRODUCTO","TIP DAY","WEEKDAY_DE DESC","ID_QUARTER C"]
	Mandatory Dimensions	[]

2	Includes	["ANYO","ID_QUARTE D_WEEK","ID_FECHA ESC","CURRENCY"]
	Mandatory Dimensions	["CURRENCY"]

3	Includes	["ANYO","ID_QUARTE D_WEEK","WEEK_DE CHA","CARTERA_DES
	Mandatory Dimensions	["CARTERA_DESC"]

Compare without / with AGGs:



Now we uses 3% more of time to build the cube and 0.6% of space, but queries by currency or Carteras_Desc will be very faster